
MASSACHUSETTS
INSTITUTE of TECHNOLOGY
LIBRARIES

A Microcomputer-Based Image Database Management System

B. E. Prasad
Amar Gupta
Hoo-min D. Toong
Stuart E. Madnick

November 1986

CISR WP No. 146
Sloan WP No. 1839-86

Center for Information Systems Research
Massachusetts Institute of Technology
Sloan School of Management
77 Massachusetts Avenue
Cambridge, Massachusetts, 02139

A Microcomputer-Based Image Database Management System

B. E. Prasad
Amar Gupta
Hoo-min D. Toong
Stuart E. Madnick

November 1986

CISR WP No. 146
Sloan WP No. 1839-86

To be published in IEEE Transactions on Industrial Electronics, February 1987.

Center for Information Systems Research
Sloan School of Management
Massachusetts Institute of Technology

A MICROCOMPUTER-BASED IMAGE DATABASE MANAGEMENT SYSTEM

Bandreddi E. Prasad, Amar Gupta,
Hoo-min D. Toong, and Stuart E. Madnick

ABSTRACT

Industrial applications frequently involve manipulation and management of pictorial information. In spite of the fact that digitized images and pictorial information are becoming more significant as parts of user databases, conventional database techniques have focused primarily on numerical and textual information. The Image Database Management System (IDBM) extends these techniques to encompass images, pictures and graphs. Designed to run in a microcomputer environment, the design of IDBM integrates efficient database algorithms and compression techniques to permit fast retrievals as well as an unusally large number of images to be set up as a single database. The support of a synonym base is another important characteristic of the IDBM system.

I. INTRODUCTION

In terms of basic computing power, the mainframes of the early 1960s, the minicomputers of the early 1970s, and the microcomputers of the 1980s are all in the same performance bracket [1]. However, each of these three decades is characterized by emphasis on a different type of information. During the 1960s, the focus was on processing numerical information. The 1970s witnessed attention geared towards processing of textual information. During the present decade increasing efforts are being concentrated towards efficient representation and manipulation of pictorial information.

To illustrate the above trend, consider the maintenance manual for any major industrial equipment. The manual typically contains engineering numbers, descriptive text, and some black-and-white or full color pictures. In the 1960s, computers were used to generate the numbers. By the late 1970s, computers began to be commonly used for processing and updating the text. Now, computers are used to create and to manipulate graphical and pictorial information [2, 3].

Even images of average complexity require large amounts of storage. A single high resolution image, in color, typically requires about a million bits of memory space. In spite of this fact, until recently little attention was paid

to the management of nonalphanumeric information. The advent of powerful low-cost microcomputers has motivated use of these systems for image-oriented applications in many different areas including interactive computer-aided design (CAD), robotics, computer-aided manufacturing (CAM) and automated process control [4, 5, 6, 28].

At this stage, it is pertinent to distinguish between two aspects of image management. One relates to the physical storage of pictorial information, and encompasses issues such as the development of new data compression techniques and error correction algorithms to enable compact and accurate representation of images. The other relates to management of image databases, and includes investigation of strategies that enable a desired sub-set of images to be retrieved based on a group of selection criteria specified by the user. The latter aspect alone is examined in this paper.

II. IMAGE DATABASE SYSTEMS

An image database is a system in which a large amount of pictorial information is stored in an integrated manner. A collection of many images does not mean it is a database. To be classified as an image database, the management aspect must also be present.

One method to extend a conventional alphanumeric database is to add images as a data type. Each image can be assigned a unique picture name, and sets of images are retrieved in a manner analogous to numbers and text. This approach has been employed in several early systems including Aggregate Data Manager [32] and Spatial Data Management System. The latter project was originally conceived at MIT [33], and subsequently a commercial model was developed at the Computer Corporation of America [34].

Instead of considering the entire picture or screen as a single indivisible unit, more sophisticated image database systems allow the image to be described in greater depth. For example, in a system called IMDS, each LANDSAT image is defined in terms of its name, the image matrix, the spectral channel, the geographic coordinates, the scale factor, the date of creation, and a legend. In cartographic applications, maps are specified in terms of primal components such as points, line segments, and polygons. Image specification strategies adopted by different designers, and instances of new languages used to specify and to retrieve pictorial information from different image databases can be divided into four categories:

1) Efforts within the framework of a relational database model [1, 9, 11, 12, 20, 21],

2) Extending/modifying relational database model to include new features for handling images [13, 17, 18],

3) Special hardware for image processing [14, 28], and

4) Specific applications [4, 5, 8, 14, 15, 20] such as medical images, LANDSAT photos and cartography.

Whereas the preferred strategy for handling alphanumeric databases is to use the relational model, the choice is not clear in the case of image databases. In [28], McKeown points out three major disadvantages of using relational database techniques. First, the use of the basic attribute-value pairs implies that all primary key attributes must be duplicated in each relation, since there is no mechanism for allowing multiple-valued entities. Second, the relational database operators (union, join, project, etc.) are ill-suited for implementing geometric notions of proximity and intersection. Third, partitioning of large image databases is difficult using the relational model. These issues become even more significant in the context of microcomputers, which inherently possess lesser computing power and smaller storage capacity than available on mainframe computers and minicomputers.

III. DESIGN OF IDBM

The Image Database Management System (IDBM) is designed to serve as a generalized tool for storing and retrieving pictorial information in a microcomputer environment. Unlike other strategies that are geared to one discipline (e.g., medical image databases, cartographic databases), IDBM employs techniques that are suitable in many different disciplines.

We distinguish between two forms of pictorial representation. A slide refers to a picture occupying the entire screen. A pix is a subset of a picture. The size and content of a pix is determined by the user by moving and scaling a pix-rectangle on the screen and enclosing the desired section of the screen in this rectangle. The difference between slides and pixes is explained in [2, 31]. The use of the dual structure allows full pictures, as well as parts of pictures, to be referenced individually. In IDBM each slide is assigned a unique slide-name. If the slide contains more than one pix, then these pixes are numbered 1, 2, 3, and so on. By concatenating the slide name and the pix number, a unique identifier (ID) is generated for each pix.

The CREATE module of IDBM enables the image to be described in terms of four sets of attributes. In the present implementation of IDBM, these attributes are termed (i)

Subjects; (ii) Emotion; (iii) Action; and (iv) Physical Attributes. In each set any desired number of elements can be specified. For example, an image may contain a computer, a table, and a man. All these three elements will be specified as subjects. The other three attributes are described in a similar manner. Further, modifiers can be used with each descriptor. For example, personal computer, mechanical part, power supply, etc. By using multiple modifier-descriptor pairs, any image can be defined with as much precision as desired.

Images pertaining to one discipline can be organized into a library. The RETRIEVE module of IDBM allows images to be retrieved in three different ways:

(i) By specifying the library number to retrieve all images in a particular discipline;

(ii) By specifying the unique identifier to retrieve a particular slide or pix; and

(iii) By specifying conditions in an attribute list to retrieve the particular set of slides that meet all the criteria.

While (i) and (ii) above are easy to implement, the process of

selective retrievals is more complex. Before discussing how images are selectively retrieved, it is necessary to first describe the basic structure of IDBM.

IV. STRUCTURE OF IDBM

IDBM consists of four main modules: (i) the specification database which contains information about each image; (ii) the syntactic database used to validate and to decompose a user query; (iii) the pictorial database which contains the images; and (iv) the user interface routines. These four modules are described in the following paragraphs.

The Specification Database

The specification database is logically organized as shown in Fig. 1. The physical structure of the specification database is shown in Fig. 2. Because a user is more likely to specify a list of attributes than the image ID's, an inverted file structure has been used. When an image is initially specified in terms of the library number, image ID, and attributes (see top left of Figs. 1 and 2), the files containing these pieces of information are updated. At Level

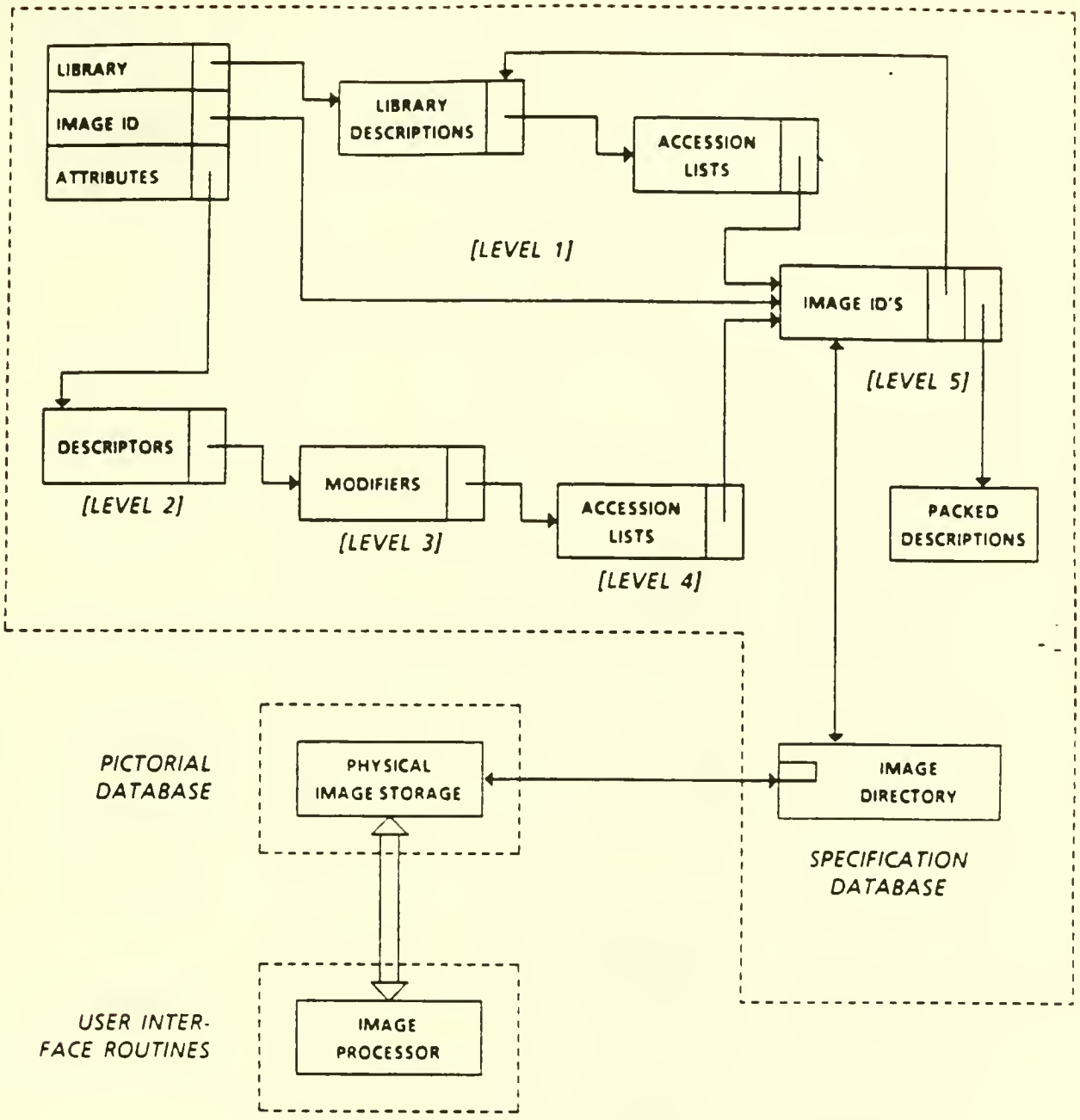

Figure 1. Logical structure of IDBM

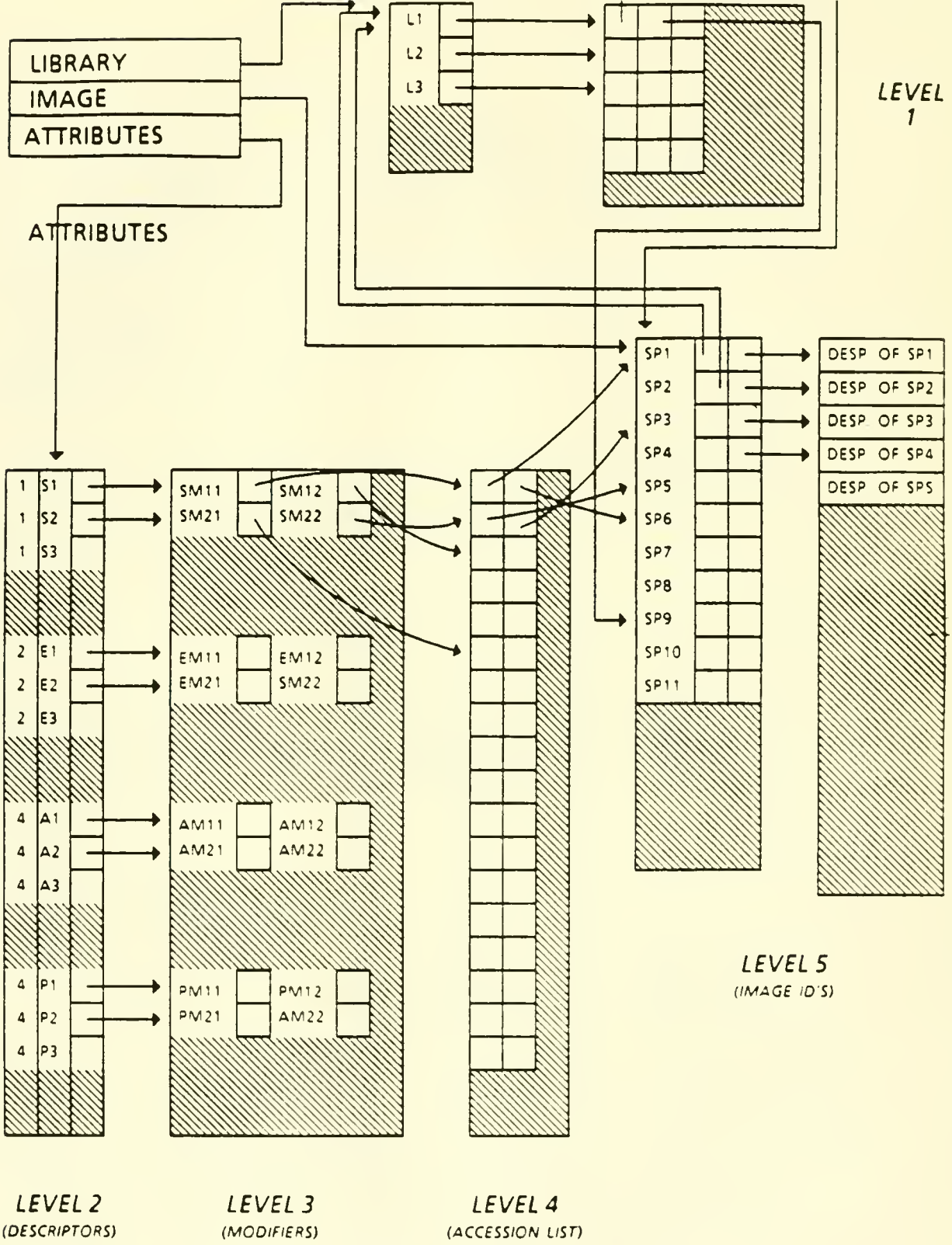

Figure 2. Physical structure of IDBM.

l the library file (middle top of figures) contains a pointer to the block (shown on top right) containing ID's of all images in that library. If the block space is insufficient to hold the information, overflow blocks are used.

The bottom left of Fig. 2 is shown in greater detail in Fig. 3. For each value of the attribute, there is a pointer indicating the beginning entry of the list of modifiers for that value. If subject-1 had four different modifiers, there would be four entries in the row in Level 3 for this subject. Each entry, in turn, contains a pointer to the block (Level 4) containing ID's of all images corresponding to that particular set of descriptor and modifier. Again, overflow blocks are automatically employed, when required.

Level 5 (see Fig. 2) contains image ID's and the corresponding physical location of the respective slides and pixes. Level 3 serves as the image directory. In Fig. 2, the image ID's have been shown as SP1, SP2, etc., to denote that an image can be either a slide or a pix. Apart from the ID and the corresponding address, there is a pointer to the location containing the full description (descriptors and modifiers) for the image. Actually, the description for any image can be assembled from the information contained in Levels 2 and 3. However, the latter process is very tedious and time consuming. As such, the descriptors are duplicated

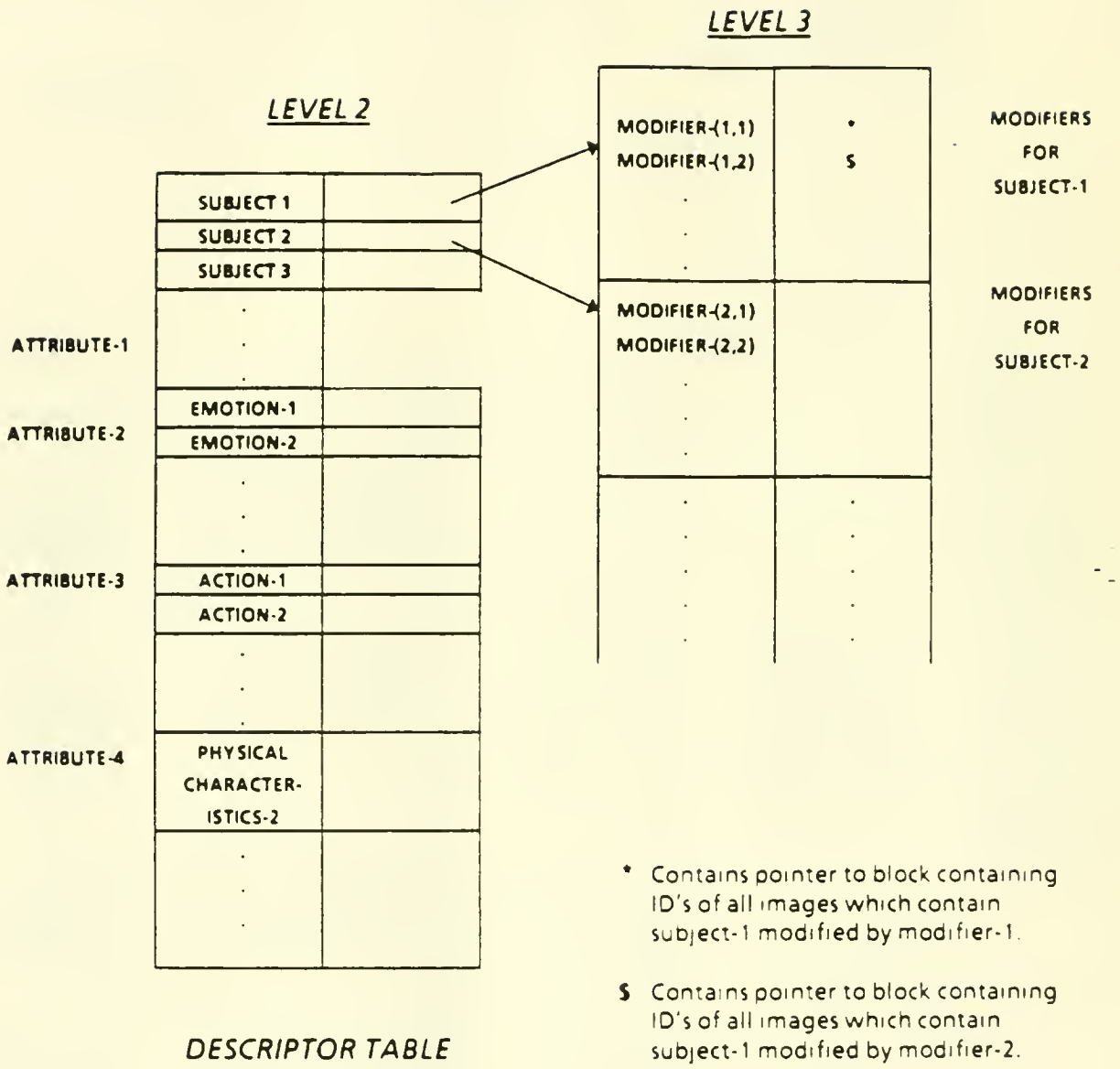

Figure 3. Storage attributes in inverted file structure.

and stored in Level 5 in the form of a packed description. This enables the full description for any image to be retrieved immediately without having to read all the entries in Levels 3 and 4.

It is appropriate to clarify one important aspect of Level 3. The right side of Fig. 3 shows the set of modifiers for subject-1, subject-2 and so on. In the parenthesis of the modifier, the first number relates to the subject, and the second to the modifier. With this convention MODIFIER (1,2) implies the second modifier relating to the first subject. The same modifier may occur under several subjects. For example, the word ELECTRONIC can be modifying COMPUTERS and COMMUNICATIONS. It may appear that the duplication of modifiers is leading to inefficiency, but this is not so. First, each modifier occupies only two bytes of storage and as such the overall contribution to program size is minimal. Second, it is essential to store modifiers, in the present form, in order to directly determine the descriptor value, to obtain efficient retrieval, and to handle accession lists with speed and accuracy.

In our opinion, the image specification structure described above offers the best response time, given the typical memory sizes available on contemporary microcomputers.

The Syntactic Database

In most cases, the person retrieving images is different from the person who initially stores the images. This makes it unrealistic to contemplate that the two persons would define a particular image in an identical manner. For example, at the initial stage, the word MEMORY may have been chosen as a subject. Later, another person could specify STORAGE as the selection criterion. For the system to be truly effective, there must be a mechanism to know that these two words are functionally equivalent.

The above objective is one of the goals of the syntactic database. This database consists of two sets of dictionaries. The entries in each of these dictionaries are of the form:

(n, w, p_s)

where

`n' is the unique number assigned to the particular word,

`w' is the word, and

`p_s' is the pointer to the synonym.

The value of p_s is zero in two cases: (i) when there is no synonym for `w'; or (ii) when `w' happens to be the word to

which all its synonyms point. The set of words characterized by p_s equal to zero are called basic words. The unique numbers associated with such words are used for internal coding. For all other words, the corresponding basic words are first assigned, and the unique numbers assigned to the latter words are then used.

As an example, assume COMPUTING and COMPUTATIONAL are desired to be stored as synonyms. Then the two entries in the dictionary may well appear as (150, COMPUTING, 0) AND (180, COMPUTATIONAL, 150). This implies that COMPUTING is a basic word, and COMPUTATIONAL is its synonym. Also, the unique number 150 will be used when storing or retrieving an image containing either of these words. Apart from mitigating the problem of functionally-equivalent words, this technique reduces the number of descriptors for each image.

Of the two dictionaries, the first dictionary contains words of universal importance. The data in this dictionary is an integral part of IDBM. The second dictionary is application-dependent, and its vocabulary is created and expanded by the user. Although both dictionaries contain the same structure of entries, the size and contents of the first dictionary are invariant whereas the second dictionary gradually grows in size. When a retrieval criterion is specified, the standard dictionary is first searched and then

the application-dependent one. If the word is located in either dictionary, then the corresponding basic word is identified and the query process proceeds. If the word is not currently in either dictionary, the user can add it to the application-dependent dictionary and also indicate its synonyms, if any.

The Pictorial Database

The image database management routines of IDBM can operate in conjunction with any pictorial database system capable of running in an IBM Personal Computer or compatible environment. Since all programming has been done in the C language, it is relatively easy to tailor IDBM routines to execute in many other computing environments. It can be used to store, catalog, and retrieve pictures, images, graphs, photographs, maps, and virtually anything that can be displayed on the screen.

Depending on the environment, the image may be drawn, generated by a digitizer, or created using another package. IDBM does not deal with the issues of creating the pictorial database. Our research has focused on identifying management strategies that can work in conjunction with state-of-the-art off-the-shelf graphics software. To demonstrate the

viability of our strategies, VCN ExecuVision was selected as a test case since it offers the largest number (over 4,000) of pre-rendered images [2]. Independent reviewers have highly recommended it as "The Cadillac of Presentation Graphics Software" [35] and "What a word processor is to words, VCN ExecuVision is to graphics" [36]. This graphics package uses sophisticated data compression algorithms to store images in a very compact mode. We used this software to design and test various routines of IDBM.

Total Storage Requirements

The total storage requirements for IDBM program routines, and the four modules (including the pictorial database) for 2,000 images are as shown in Table 1. Note that the total storage requirements of 8 Mbytes fall within the storage capacity of contemporary hard disks. IDBM program routines account for only 84 Kbytes. The rest is comprised of data.

V. USING IDBM

When a user enters a set of selection criteria, the sequence of steps performed by IDBM is summarized in Fig. 4.

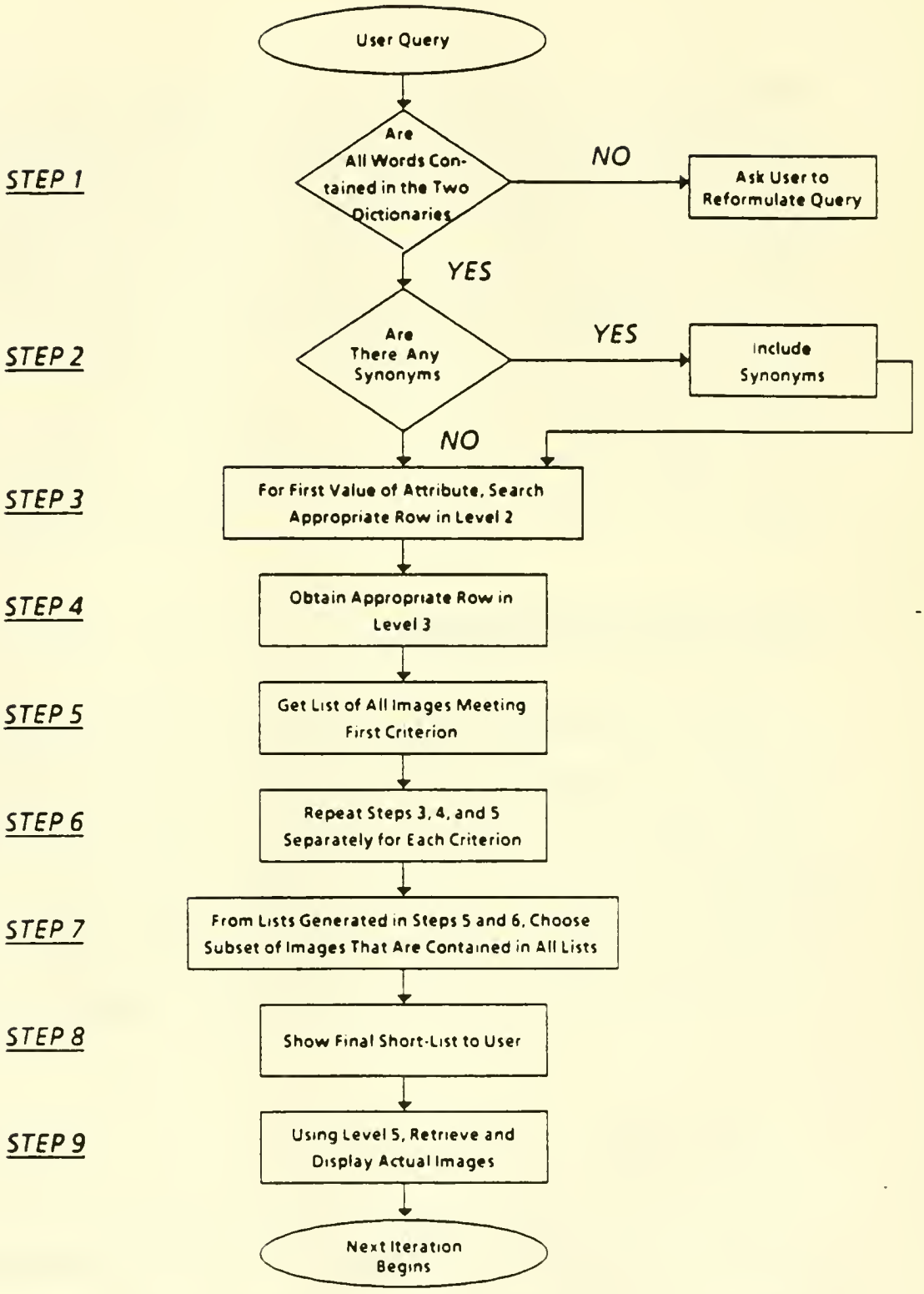

Figure 4. Sequence of steps in response to user query.

ENTITY	SIZE/ASSUMPTION	STORAGE IN KBYTES
Standard Dictionary	10,000 words	300
User Dictionary	2,000 words	50
Level 1	25 libraries	9
Level 2	4,000 attribute values	28
Level 3	2 modifier/attribute values	128
Level 4	10 attribute values/ID	256
Level 5	2,000 IDs	208
Hash Overhead	----	100
	Total =	1,079 (i)

Average storage requirement per image = 3.5 Kbytes
Storage requirement for 2,000 images = 2,000 x 3.5 = 7,000 Kbytes (ii)
Total storage requirements = (i) + (ii)
= 1,079,000 + 7,000,000 bytes
= 8 Mbytes

Table I. Storage Requirements For an IDBM System With 2,000 Images

The general form of a query based on attributes is as follows:

$$A_1\ (m_1, v_1)\ \&\ A_2\ (m_2, v_2)\ \&\ .....\ \&\ A_k\ (m_k, v_k)$$

where,

A_i is one of the attributes: SUBJECT, ACTION, EMOTION, PHYSICAL CHARACTERISTICS;

m_i is a modifier; and

v_i is an attribute value also known as descriptors.

Suppose a user wanted to retrieve all images that pertained to the subject "CAD/CAM APPLICATIONS ON PERSONAL COMPUTERS." Under an IDBM environment, this requirement would be translated as "show all images containing subjects <CAD;CAM;application;personal computer>." The word `personal' is a modifier to the descriptor `computer,' while the other descriptors carry no modifiers. This user query can be written formally as:

$$A_1\ (@,CAD)\ \&\ A_2\ (@,CAM)\ \&\ A_3\ (@,application)\ \&\ A_4\ (personal, computer)$$

where @ refers to null string and

A_1, A_2, A_3, and A_4 are all SUBJECTS in this case.

Based on the above query, IDBM first searches for all images containing CAD and its synonyms (computerized design; automated design, etc.) and generates a first list. Similarly, three lists are generated for the other three descriptors. Finally, a short-list is compiled containing image ID's in all four lists. From a technical perspective, the process can be visualized in terms of four steps as follows:

Step 1: Using the ordered pairs (m_1, v_1), (m_2, v_2) ..... (m_k, v_k), search the standard dictionary to get the corresponding numbers. Ignore any m_i which is a null string or a blank. If a word is not found in the standard dictionary, search the application-dependent dictionary. If the word is still not found, trigger an error message. If Step 1 is completed properly, assume that the number pairs obtained for the query are

$$(nm_1, nv_1), (nm_2, nv_2) \ldots\ldots (nm_k, nv_k)$$

where

$nm_i = 0$ if m_i = @(null string).

Step 2: Depending on attribute A_i associated with (m_i, v_i),

search the corresponding attribute file for the number nv_i. Then,

if $nm_i = 0$ take all pointers to the accession list associated with modifiers of nv_i;
if $nm_i > 0$ take only the pointer associated with nm_i.

Repeat this step for all the pairs in the query.

Step 3: Accession lists obtained by Step 2 contain pointers to slide data file. Assume that B_1, B_2 ..... B_k are the accession lists for the given query. Since each query condition is connected by an `and' operation, we select those pointers which appear in all the accession lists. Denote this set by B*. The number of values in B* is equivalent to the total number of images (slides and pixes) that meet the criteria specified by the user.

Step 4: Using the pointers in B*, retrieve the slide names and pix numbers which are then passed to the IMAGE SYSTEM for display.

After completing Step 3, the total number of images that meet all the selection criteria is displayed. On request, the number of images fulfilling individual criterion is also

displayed. This enables the user to reframe his or her query to come up with the appropriate number of images. These images are then displayed using the pictorial database and the interface routines.

Instead of retrieving information, if the user desired to add an image to the database, a template is provided to enter the specifications for the particular image. If the user entered a descriptor that is not contained in either of the two dictionaries, the user is informed accordingly. There are two possibilities: (i) either the user specified an incorrect spelling and as such he or she can now state the correct word; or (ii) the user wishes to enter a new word in the dictionary. In the latter case, IDBM provides the facility for specifying synonyms as well, as described in the previous section.

It is clarified that the current version of IDBM supports only the logical `AND' between values specified in the retrieval criteria. A user desiring to specify an `OR' operation must use a succession of queries. For example, to get all images containing either a CAR or a TRUCK, the user first obtains all images specifying CAR alone. Then, he or she can retrieve images containing TRUCK. As this process can get cumbersome, we are currently enhancing the logical set to allow for operations other than a simple `AND'.

The retrieval speed and efficiency of IDBM is heavily influenced by (i) the size of the syntactic database; (ii) the level of specification of each image at the time of storage; and (iii) the level of detail of the retrieval criteria. Using a sub-set of VCN ExecuVision libraries containing 400 (of the 4,000) images, and specifying each image in terms of an average of 10 descriptor-modifier pairs, the observed response time has been 5 seconds or less using an IBM PC/XT system. The performance will be still better using a faster computer. This response time is acceptable for most applications.

VI. UNIQUE FEATURES AND NEW DIRECTIONS

As explained earlier, the subject of image database management system has received attention of several researchers in recent years. In our opinion, the unique features of IDBM are as follows:

(1) Whereas almost all other image database management systems have been hosted only on mainframes or minicomputers, IDBM has been specifically developed to work in an IBM Personal Computer (or compatible) environment;

(2) IDBM has been designed for storing and retrieving icons, symbols, and images, unlike other microcomputer-based

database packages (e.g., dBase-II) which are directed towards numeric and textual information;

(3) IDBM allows multiple values per attribute to be specified at the time of storage and also at the time of retrieval;

(4) IDBM supports the mechanism for automatic checking for synonyms. This eliminates the need for the user to be aware of what particular descriptors have been used previously in the CREATE mode.

It is pertinent to mention here that IDBM can be used in conjunction with any pictorial information database. This includes the spectrum of areas from maps used in cartographic applications to graphs used in industrial applications and from spreadsheets used in the office environment to photographic information gathered by satellites. The first version of IDBM has been tested in conjunction with VCN ExecuVision [2, 31] presentation graphics software. This package was preferred because it offered immediate access to several thousand prerendered professional quality images, and because it is widely used in industrial and business applications.

The concept of IDBM was originally conceived to encapsulate the intelligence by which a user is able to selectively recall the images that pertain to a particular

topic of interest. However, in most environments, information of many different types, besides images, is needed, and no distinction is made between numeric, textual, graphical and pictorial information. IDBM has recently been enlarged to allow for storage and retrieval of numeric and textual information. At present, new input and output routines are being developed to allow IDBM to operate in conjunction with the software packages commonly used for manipulating numbers and text in a microcomputer environment.

REFERENCES

1. H. D. Toong and A. Gupta, "Personal Computers," Scientific American, vol. 247, no. 6, Dec. 1982, pp. 88-99.

2. H. D. Toong and A. Gupta, "A New Direction in Personal Computer Software," Proc. of the IEEE, VOL. 72, NO. 3, March 1984, pp. 377-388.

3. S. Feiner, S. Nagy and A. Van Dam, "An experimental system for creating and presenting graphical documents," ACM Trans. Graphics, vol. 2, no. 1, January 1982, pp. 59-77.

4. J. C. Dorng and S. K. Chang, "Design considerations for CAD/CAM databases," in Proc. Int. Computer Symp., Taipei, Taiwan, December 1984.

5. Y-C. Lee and K. S. Fu, "A CAD/CAM database management system and its query languages," in Languages for Automation, edited by S. K. Chang, Plenum: New York, 1985.

6. S. K. Chang, "Image Information Systems," Proc. of the IEEE, vol. 73, no. 4, April 1985, pp. 754-764.

7. R. B. Abhyankar and R. L. Kashyap, "Pictorial Data Description and Retrieval with Relational Languages," IEEE Picture Data Description & Mgmt. Workshop, 1980, pp. 57-60.

8. T. Ichikawa, T. Kikuno and M. Hirakawa, "A Query Manipulation System for Image Data Retrieval by ARES," IEEE Picture Data Description & Mgmt. Workshop, 1980, pp. 61-67.

9. N. S. Chang and K. S. Fu, "A Query Language for Relational Image Database Systems," IEEE Picture Data Description & Mgmt. Workshop, 1980, pp. 68-73.

10. S. Levialdi, "Programming in PIXAL," IEEE Picture Data Description & Mgmt. Workshop, 1980, pp. 74-82.

11. B. S. Lin and S. K. Chang, "GRAIN--A Pictorial Database Interface," IEEE Picture Data Description & Mgmt. Workshop, 1980, pp. 83-88.

12. S. Uno and H. Matsuka, "A Relational Database for Design Aids System," IEEE Picture Data Description & Mgmt. Workshop, 1980, pp. 89-94.

13. K. Yamaguchi et al., "ELF: Extended Relational Model for Large, Flexible Picture Database," IEEE Picture Data Description & Mgmt. Workshop, 1980, pp. 95-102.

14. M. Friedell et al., "The Management of Very Large Two-dimensional Raster Graphics Env.," IEEE Picture Data Description & Mgmt. Workshop, 1980, pp. 139-144.

15. D. M. McKeown, Jr., "Knowledge Structuring in Task Oriented Image Databases," IEEE Picture Data Description & Mgmt. Workshop, 1980, pp. 145-151.

16. P. G. Selfridge, "Name-Value Slots and the Storage of Image Information," IEEE Picture Data Description & Mgmt. Workshop, 1980, pp. 152-157.

17. G. Y. Tang, "A Logical Data Organization for the Integrated Databases of Pictures and Alphanuemrical Data," IEEE Picture Data Description & Mgmt. Workshop, 1980, pp. 158-166.

18. G. Y. Tang, "A Management System for the Integrated Databases of Pictures and Alphanumerical Data," Comput. Graphics & Image Process., vol. 16, 1981, pp. 270-286.

19. K. R. Sloan, Jr. and A. Lippman, "Databases of/about/with images," Proc. of IEEE Computer Society Conf. on Pattern Recognition and Image Processing, 1982, pp. 441-446.

20. M. Nagata, "A relational image data base system for remote sensing (LAND DBMS)," Proc. of IEEE Computer Society Conf. on Pattern Recognition and Image Processing, 1982, pp. 491-495.

21. N. S. Chang and K. S. Fu, "Query-by-pictorial-example," IEEE Trans. Softw. Engg., SE-6, 1980, pp. 519-524.

22. N. S. Chang and K. S. Fu, "Picture query languages for pictorial database systems," IEEE Computer, vol. 14, no. 11, 1981.

23. S. K. Chang and K. S. Fu, "Pictorial data base systems," Springer-Verlag: Berlin, FRG, 1980.

24. S. K. Chang and T. L. Kunii, "Pictorial Data base Systems," IEEE Computer, vol. 14, no. 11, 1981.

25. A. Blaser (ed.), "Data base techniques for pictorial applications," Springer-Verlag: Berlin, FRG, 1980.

26. H. Tamura and N. Yokoya, "Image database systems: A

survey," Pattern Recognition, vol. 17, no. 1, January 1984.

27. G. Nagy, "Image Database," Image and Vision Computing, vol. 3, no. 3, August 1985.

28. D. M. McKeown, Jr., "Digital catography and photo interpretation from a data base view point," in New Applications of Databases, Academic Press, 1984, pp. 19-42.

29. M. Crehange et al., "Exprim: an expert system to aid in progressive retrieval from a pictorial and descriptive data base," in New Applications of Databases, Academic Press, 1984, pp. 19-42.

30. S. K. Chang and S. H. Liu, "Picture indexing and abstraction techniques for pictorial databases," IEEE Trans. Pattern Anal. Mach. Intell., vol. 6, no. 4, 1984.

31. A. Gupta and H. D. Toong (eds.), Insights into Personal Computers, IEEE Press: New York, 1985.

32. Y. Takao, S. Itoh and J. Iisaka, "An image-oriented database system," in Data Base Techniques for Pictorial Applications, A. Blaser, ed., Springer-Verlag, 1980, pp. 527-538.

33. W. C. Donelson, "Spatial management of information," Computer Graphics, vol. 12, 1978, pp. 203-209.

34. C. F. Herot, "A prototype spatial data management system," Computer Graphics, vol. 14, 1980, pp. 63-70.

35. _____, "Great and `Not-So-Great Graphics'," PC, June 11, 1985, pp. 160-161.

36. W. J. Hawkins, "Bits and bytes," Popular Science, January 1984.